RKNN-Toolkit2 API 参考手册

文件标识：RK-YH-YF-412

发布版本：V2.4.0

日期：2026-1-15

文件密级：□绝密　□秘密　□内部资料　■公开

免责声明

本文档按"现状"提供，瑞芯微电子股份有限公司（"本公司"，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

"Rockchip"、"瑞芯微"、"瑞芯"均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2024 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：　福建省福州市铜盘路软件园A区18号

网址：　www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

前言

概述

本文是 RKNN-Toolkit2 的 API 参考手册。

RKNN-Toolkit2 是为用户提供在PC平台上进行模型转换、推理和性能评估的开发套件。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明	核定人
V1.6.0	HPC团队	2023-11-15	初始版本	熊伟
V2.0.0-beta0	HPC团队	2024-03-22	1. 增加RK3576相关描述 2. 增加2.2章节的sparse_infer推理接口说明 3. 更新2.7章节的core_mask参数说明 4. 增加2.9章节的eval_perf的fix_freq参数说明 5. 更新2.16章节的自定义算子用法说明	熊伟
V2.1.0	HPC团队	2024-07-29	1. 更新2.2章节的quantized_dtype参数说明 2. 增加2.2章节的enable_flash_attention参数说明 3. 增加2.7章节的fallback_prior_device参数说明 4. 移除2.5章节的cpp_gen_cfg参数说明 5. 增加2.17章节的生成C++部署示例说明	熊伟
V2.2.0	HPC团队	2024-09-04	1. 增加RV1106B相关描述 2. 增加1.2章节的Python3.12说明	熊伟
V2.3.0	HPC团队	2024-11-04	1. 增加ARM64版本说明 2. 更新1.3章节的深度学习框架的版本说明 3. 更新2.2章节的quantized_dtype参数说明	熊伟
V2.3.2	HPC团队	2025-4-1	1. 增加自动混合量化功能说明 2. 增加分组量化的参数说明 3. 增加GDQ的量化算法说明 4. 增加RV1126B相关描述	熊伟
V2.4.0	HPC团队	2026-1-15	1. 支持mini-runtime转换设置和连板调试 2. 修复精度分析因科学计数法导致的显示异常问题 3. 支持qat模型开启稀疏化推理功能 4. 增加disable_first_conv和enable_core0_only_mode的配置项	熊伟

目录

RKNN-Toolkit2 API 参考手册
- 1 要求
  - 1.1 适用芯片
  - 1.2 系统依赖说明
  - 1.3 适用的深度学习框架
- 2 API详细说明
  - 2.1 RKNN初始化及释放
  - 2.2 模型配置
  - 2.3 模型加载
    - 2.3.1 Caffe模型加载接口
    - 2.3.2 TensorFlow模型加载接口
    - 2.3.3 TensorFlow Lite模型加载接口
    - 2.3.4 ONNX模型加载
    - 2.3.5 DarkNet模型加载接口
    - 2.3.6 PyTorch模型加载接口
  - 2.4 构建RKNN模型
  - 2.5 导出RKNN模型

2.6 加载RKNN模型
2.7 初始化运行时环境
2.8 模型推理
2.9 评估模型性能
2.10 获取内存使用情况
2.11 查询SDK版本
2.12 混合量化
    2.12.1 hybrid_quantization_step1
    2.12.2 hybrid_quantization_step2
2.13 量化精度分析
2.14 获取设备列表
2.15 导出加密模型
2.16 注册自定义算子
2.17 生成C++部署示例

1 要求

1.1 适用芯片

RKNN-Toolkit2当前版本所支持芯片的型号如下：

- RV1103
- RV1103B
- RV1106
- RV1106B
- RV1126B
- RK2118
- RK3562
- RK3566系列
- RK3568系列
- RK3576系列
- RK3588系列

注：后文用RK3566 / RK3568 / RK3576 / RK3588分别统称RK3566系列 / RK3568系列 / RK3576系列 / RK3588系列。

1.2 系统依赖说明

使用RKNN-Toolkit2时需要满足以下运行环境要求：

操作系统版本	**Ubuntu18.04（x64）**	**Ubuntu20.04（x64）**	**Ubuntu22.04（x64）**	**Ubuntu24.04（x64）**
Python版本	3.6 / 3.7	3.8 / 3.9	3.10 / 3.11	3.12

ARM64版本运行环境要求：

操作系统版本	**Debian10（x64)**	**Debian11（x64)**	**Debian12（x64)**
Python版本	3.6 / 3.7	3.8 / 3.9	3.10 / 3.11 / 3.12

注：

1. 具体python库依赖详见doc/requirements*.txt
2. 本文档主要以Ubuntu 20.04 / Python3.8为例进行说明

1.3 适用的深度学习框架

RKNN-Toolkit2支持的深度学习框架包括Caffe、TensorFlow、TensorFlow Lite、ONNX、DarkNet和PyTorch。

它和各深度学习框架的版本对应关系如下：

RKNN-Toolkit2	Caffe	TensorFlow	TF Lite	ONNX	DarkNet	PyTorch
1.4.0 1.4.2 1.5.0 1.5.2	1.0	1.12.0~ 2.8.0	Schema version=3	1.7.0~ 1.10.0	Commit ID: 810d7f7	1.6.0~ 1.10.1
1.6.0	1.0	1.12.0~ 2.14.0	Schema version=3	1.7.0~ 1.14.0	Commit ID: 810d7f7	1.6.0~ 1.13.1
2.0.0 2.1.0 2.2.0	1.0	1.12.0~ 2.14.0	Schema version=3	1.7.0~ 1.14.1	Commit ID: 810d7f7	1.10.1~ 2.1.0
2.3.0 2.3.2	1.0	1.12.0~ 2.14.0	Schema version=3	1.7.0~ 1.17.0	Commit ID: 810d7f7	1.10.1~ 2.4.0

注：

1. 由于TensorFlow版本兼容特性，TensorFlow 1.12.0之前版本的导出的pb文件，理论上也是支持的。关于TensorFlow版本兼容性的更多信息，可以参考官方资料：https://www.tensorflow.org/guide/versions?hl=zh-cn
2. 因为TFLite不同版本的schema之间是互不兼容的，所以构建TFLite模型时使用与RKNN-Toolkit2不同版本的schema可能导致加载失败。
3. RKNN-Toolkit2使用的Caffe protocol是基于berkeley官方修改的protocol：https://github.com/BVLC/caffe/tree/master/src/caffe/proto，commit值为828dd10，RKNN-Toolkit2在此基础上新增了一些OP。
4. ONNX release version和opset version、IR version之间的关系参考onnxruntime官网说明：https://github.com/microsoft/onnxruntime/blob/v1.6.0/docs/Versioning.md
5. DarkNet官方Github链接：https://github.com/pjreddie/darknet。RKNN-Toolkit2现在的转换规则是基于master分支的最新提交（commit值：810d7f7）制定的。
6. 加载PyTorch模型（torchscript模型）时，推荐使用相同版本的PyTorch导出模型并转为RKNN模型，前后版本不一致时有可能会导致转RKNN模型失败。
7. ARM64版本仅支持PyTorch和ONNX框架，其他框架暂时不支持。

2 API详细说明

2.1 RKNN初始化及释放

在使用RKNN-Toolkit2的所有API接口时，都需要先调用RKNN()方法初始化RKNN对象，不再使用该对象时通过调用该对象的release()方法进行释放。

初始化RKNN对象时，可以设置verbose和verbose_file参数，以打印详细的日志信息。其中verbose参数指定是否要打印详细日志信息；如果设置了verbose_file参数，且verbose参数值为True，日志信息还将写到该参数指定的文件中。

举例如下：

```
# 打印详细的日志信息
rknn = RKNN(verbose=True)

…

rknn.release()
```

2.2 模型配置

在构建RKNN模型之前，需要先对模型进行通道均值、量化图片RGB2BGR转换、量化类型等的配置，这些操作可以通过config接口进行配置。

API	config
描述	设置模型转换参数。
参数	**mean_values**：输入的均值。参数格式是一个列表，列表中包含一个或多个均值子列表，多输入模型对应多个子列表，每个子列表的长度与该输入的通道数一致，例如[[128,128,128]]，表示一个输入的三个通道的值减去128。 默认值为None，表示所有的mean值为0。
	std_values：输入的归一化值。参数格式是一个列表，列表中包含一个或多个归一化值子列表，多输入模型对应多个子列表，每个子列表的长度与该输入的通道数一致，例如[[128,128,128]]，表示设置一个输入的三个通道的值减去均值后再除以128。 默认值为None，表示所有的std值为1。
	quant_img_RGB2BGR：表示在加载量化图像时是否需要先做RGB2BGR的操作。如果有多个输入，则用列表包含起来，如[True, True, False]。默认值为False。 该配置一般用在Caffe的模型上，Caffe模型训练时大多会先对数据集图像进行RGB2BGR转换，此时需将该配置设为True。 另外，该配置只对量化图像格式为jpg/png/bmp有效，npy格式读取时会忽略该配置，因此当模型输入为BGR时，npy也需要为BGR格式。 **该配置仅用于在量化阶段（build接口）读取量化图像或量化精度分析（accuracy_analysis接口），并不会保存在最终的RKNN模型中，因此如果模型的输入为BGR，则在调用toolkit2的inference或C-API的run函数之前，需要保证传入的图像数据也为BGR格式。**
	quantized_dtype：量化类型，目前支持的量化类型有w8a8、w4a16、w8a16、w4a8、w16a16i和w16a16i_dfp。默认值为w8a8。 **- w8a8**：权重为8bit非对称量化精度，激活值为8bit非对称量化精度。**（RK2118不支持）** **- w4a16**：权重为4bit非对称量化精度，激活值为16bit浮点精度。**（仅RK3576/RV1126B支持）** **- w8a16**：权重为8bit非对称量化精度，激活值为16bit浮点精度。**（仅RK3562支持）** **- w4a8**：权重为4bit非对称量化精度，激活值为8bit非对称量化精度。**（暂不支持）** **- w16a16i**：权重为16bit非对称量化精度，激活值为16bit非对称量化精度。**（仅RV1103/RV1106支持）** **- w16a16i_dfp**：权重为16bit动态定点量化精度，激活值为16bit动态定点量化精度。**（仅RV1103/RV1106支持）**
	quantized_algorithm：计算每一层的量化参数时采用的量化算法，目前支持的量化算法有：normal，mmse，kl_divergence及gdq。默认值为normal。 **normal**量化算法的特点是速度较快，推荐量化数据集量一般为20-100张左右，更多的数据集量下精度未必会有进一步提升。 **mmse**量化算法由于采用暴力迭代的方式，速度较慢，但通常会比normal具有更高的精度，推荐量化数据集量一般为20-50张左右，用户也可以根据量化时间长短对量化数据集量进行适当增减。 **kl_divergence**量化算法所用时间会比normal多一些，但比mmse会少很多，在某些场景下（feature分布不均匀时）可以得到较好的改善效果，推荐量化数据集量一般为20-100张左右。 **gdq**量化算法只在w4a16和w8a16下有效，能有效的提高w4a16和w8a16的权重量化精度，推荐量化数据集量为200张以上。
	quantized_method：目前支持layer，channel或者group{SIZE}。默认值为channel。 - layer：每层的weight只有一套量化参数； - channel：每层的weight的每个输出通道都有一套量化参数，通常情况下channel会比layer精度更高。 - group{SIZE}：在channel的基础上，每个输出通道的weight在输入通道上又按{SIZE}细分为多个小组，每个小组有各自的一套量化参数，通常情况下group{SIZE}会比channel精度更高。{SIZE}是32到256之间的32的倍数值，如group32或group128。目前group{SIZE}只在quantized_dtype为w4a16下有效。
	float_dtype：用于指定非量化情况下的浮点的数据类型，目前支持的数据类型有float16。默认值为float16。
	optimization_level：模型优化等级。默认值为3。 通过修改模型优化等级，可以关掉部分或全部模型转换过程中使用到的优化规则。该参数的默认值为3，打开所有优化选项。值为2或1时关闭一部分可能会对部分模型精度产生影响的优化选项，值为0时关闭所有优化选项。
	target_platform：指定RKNN模型是基于哪个目标芯片平台生成的。目前支持“rv1103”、“rv1103b”、“rv1106”、“rv1106b”、“rv1126b”、“rk2118”、“rk3562”、“rk3566”、“rk3568”、“rk3576”和“rk3588”。该参数对大小写不敏感。默认值为None。
	custom_string：添加自定义字符串信息到RKNN模型，可以在runtime时通过query查询到该信息，方便部署时根据不同的RKNN模型做特殊的处理。默认值为None。
	remove_weight：去除conv等权重以生成一个RKNN的从模型，该从模型可以与带完整权重的RKNN模型共享权重以减少内存消耗。默认值为False。
	compress_weight：压缩模型权重，可以减小RKNN模型的大小。默认值为False。
	single_core_mode：是否仅生成单核模型，可以减小RKNN模型的大小和内存消耗。默认值为False。目前仅对RK3588 / RK3576生效。默认值为False。
	model_pruning：对模型进行无损剪枝。对于权重稀疏的模型，可以减小转换后RKNN模型的大小和计算量。默认值为False。
	op_target：用于指定OP的具体执行目标（可以设置运行在NPU/CPU/GPU等），通过输出tensor名的方式，{'op0_output_name':'cpu', 'op1_output_name':'npu', ...}， 或者通过指定op_type的方式，{'op_type':'cpu'....}， 同时还可以支持混合使用，{'op_type':'cpu', 'op0_output_name':'cpu'....} 。默认值为None。 其中，'op0_output_name'和'op1_output_name'为对应OP的输出tensor名，可以通过精度分析（accuracy_analysis）功能的返回结果中获取。'cpu'和'npu'则表示该tensor对应的OP的执行目标是CPU或NPU，目前可选的选项有：'cpu' / 'npu' / 'gpu' / 'auto'，其中，'auto'是自动选择执行目标。指定op_type的方式，同样需要通过精度分析返回结果中获取，可以将模型内所有该op_type的执行目标设置到CPU或NPU，例如，{'Add':'cpu', 'Expand':'cpu',....}。
	dynamic_input：用于根据用户指定的多组输入shape，来模拟动态输入的功能。格式为[[input0_shapeA, input1_shapeA, ...], [input0_shapeB, input1_shapeB, ...], ...]。 默认值为None，实验性功能。 假设原始模型只有一个输入，shape为[1,3,224,224]，或者原始模型的输入shape本身就是动态的，如shape为[1,3,height,width]或[1,3,-1,-1]，但部署的时候，需要该模型支持3种不同的输入shape，如[1,3,224,224], [1,3,192,192]和[1,3,160,160]，此时可以设置dynamic_input=[[[1,3,224,224]], [[1,3,192,192]], [[1,3,160,160]]]，转换成RKNN模型后进行推理时，需传入对应shape的输入数据。 注： **1. 需要原始模型本身支持动态输入才可开启此功能，否则会报错。** **2. 如果原始模型输入shape本身就是动态的，则只有动态的轴可以设置不同的值。**
	quantize_weight：在build接口的do_quantization为False情况下, 通过对一些权重进行量化以减小rknn模型的大小。 默认值为False。

API	config
	remove_reshape：删除模型的输入和输出中可能存在的Reshape的OP，以提高模型运行时性能（因为目前很多平台的Reshape是跑在cpu上，相对较慢）。 默认为 False。 注：**开启后可能会修改模型的输入或输出节点的shape，需要留意观察转换过程中的warning打印，并在部署时也需要考虑输入和输出shape变化的影响。**
	sparse_infer：在已经稀疏化过的模型上进行稀疏化推理，以提高推理性能。 目前仅对RK3576/RV1126B生效。默认为 False。
	enable_flash_attention：是否启用Flash Attention。默认为 False。 注：FlashAttention是基于 https://arxiv.org/abs/2307.08691 实现, 通过高速缓存内循环实现加速以及减少宽带使用，但是会导致模型增大，请根据具体场景和模型选择是否开启使用。更多详情请查看“RKNN Compiler Support Operator List”的exSDPAttention说明。
	auto_hybrid_cos_thresh：模型量化时开启自动混合量化中余弦距离的阈值。默认为 0.98。
	auto_hybrid_euc_thresh：模型量化时开启自动混合量化中欧式距离的阈值。默认为 None，即默认不启用。
	memory_plan_strategy：指定编译 rknn 模型过程中 Internal 内存的分配方案，影响最终模型上板推理的内存占用。可选项为 BestFit、GreedyBySize、GreedyByLifetime、SlowPeakGrowth。默认为 BestFit *BestFit* - 所有任务根据起始时间排序，起始时间一致时，内存大者优先。随后依次分配任务内存地址，在已分配内存的地址中，为当前任务寻找大小最接近的一块空闲内存进行分配。 *GreedyBySize* - 所有任务根据所需内存大小排序，内存大小一致时，起始时间前者优先。随后依次分配任务内存地址，在已分配内存的地址中，为当前任务寻找大小最接近的一块空闲内存进行分配。 *GreedyByLifetime* - 所有任务根据起始时间排序，起始时间一致时，内存大者优先。搜集每一个已分配内存的终止地址作为候选(额外包含0地址)，寻找生命周期最长任务和低位无冲突的候选地址进行分配，循环遍历直到分配完成 *SlowPeakGrowth* - 所有任务根据起始时间排序，起始时间一致时，内存大者优先。搜集每一个已分配内存的终止地址作为候选(额外包含0地址)，寻找使内存峰值增长最小的任务和低位无冲突的候选地址进行分配，循环遍历直到分配完成
	compute_order：指定 rknn 推理时图的执行拓扑顺序，部分模型推理时的 Internal 内存占用会受此参数影响，尤其是带有多分支结构的模型。可选项为 WidthFirst、DepthFrist。默认为 WidthFirst。
返回值	无。

举例如下：

```
# model config
rknn.config(mean_values=[[103.94, 116.78, 123.68]],
            std_values=[[58.82, 58.82, 58.82]],
            quant_img_RGB2BGR=True,
            target_platform='rk3566')
```

2.3 模型加载

RKNN-Toolkit2目前支持Caffe、TensorFlow、TensorFlow Lite、ONNX、DarkNet、PyTorch等模型的加载转换，这些模型在加载时需调用对应的接口，以下为这些接口的详细说明。

2.3.1 Caffe模型加载接口

API	load_caffe
描述	加载Caffe模型。（ARM64版本暂不支持该接口）
参数	**model**：Caffe模型文件（.prototxt后缀文件）路径。
	blobs：Caffe模型的二进制数据文件（.caffemodel后缀文件）路径。
	input_name：Caffe模型存在多输入时，可以通过该参数指定输入层名的顺序，形如['input1','input2','input3']，注意名字需要与模型输入名一致；默认值为None，表示按Caffe模型文件（.prototxt后缀文件）自动给定。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前路径加载mobilenet_v2模型
ret = rknn.load_caffe(model='./mobilenet_v2.prototxt',
                      blobs='./mobilenet_v2.caffemodel')
```

2.3.2 TensorFlow模型加载接口

API	load_tensorflow
描述	加载TensorFlow模型。（ARM64版本暂不支持该接口）
参数	**tf_pb**：TensorFlow模型文件（.pb后缀）路径。
	inputs：模型的输入节点（tensor名），支持多个输入节点。所有输入节点名放在一个列表中。
	input_size_list：每个输入节点对应的shape，所有输入shape放在一个列表中。如示例中的ssd_mobilenet_v1模型，其输入节点对应的输入shape是[[1, 300, 300, 3]]。
	outputs：模型的输出节点（tensor名），支持多个输出节点。所有输出节点名放在一个列表中。
	input_is_nchw：模型的输入的layout是否已经是NCHW。默认值为False，表示默认输入layout为NHWC。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载ssd_mobilenet_v1_coco_2017_11_17模型
ret = rknn.load_tensorflow(tf_pb='./ssd_mobilenet_v1_coco_2017_11_17.pb',
                           inputs=['Preprocessor/sub'],
                           outputs=['concat', 'concat_1'],
                           input_size_list=[[300, 300, 3]])
```

2.3.3 TensorFlow Lite模型加载接口

API	load_tflite
描述	加载TensorFlow Lite模型。（ARM64版本暂不支持该接口）
参数	**model**：TensorFlow Lite模型文件（.tflite后缀）路径。
	input_is_nchw：模型的输入的layout是否已经是NCHW。默认值为False，即默认输入layout为NHWC。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载mobilenet_v1模型
ret = rknn.load_tflite(model='./mobilenet_v1.tflite')
```

2.3.4 ONNX模型加载

API	load_onnx
描述	加载ONNX模型。
参数	**model**：ONNX模型文件（.onnx后缀）路径。
	inputs：模型输入节点（tensor名），支持多个输入节点，所有输入节点名放在一个列表中。默认值为None，表示从模型里获取。
	input_size_list：每个输入节点对应的shape，所有输入shape放在一个列表中。如inputs有设置，则input_size_list也需要被设置。默认值为None。
	input_initial_val：设置模型输入的初始值，格式为ndarray的列表。默认值为None。主要用于将某些输入固化为常量，对于不需要固化为常量的输入可以设为None，如[None, np.array([1])]。
	outputs：模型的输出节点（tensor名），支持多个输出节点，所有输出节点名放在一个列表中。默认值为None，表示从模型里获取。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载arcface模型
ret = rknn.load_onnx(model='./arcface.onnx')
```

2.3.5 DarkNet模型加载接口

API	load_darknet
描述	加载DarkNet模型。（ARM64版本暂不支持该接口）
参数	**model**：DarkNet模型文件（.cfg后缀）路径。
	weight：权重文件（.weights后缀）路径。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载yolov3-tiny模型
ret = rknn.load_darknet(model='./yolov3-tiny.cfg',
                        weight='./yolov3.weights')
```

2.3.6 PyTorch模型加载接口

API	load_pytorch
描述	加载PyTorch模型。 支持量化感知训练（QAT）模型，但需要将torch版本更新至1.9.0以上。
参数	**model**：PyTorch模型文件（.pt后缀）路径，而且需要是torchscript格式的模型。
	input_size_list：每个输入节点对应的shape，所有输入shape放在一个列表中。
返回值	0：导入成功。
	-1：导入失败。

举例如下：

```
# 从当前目录加载resnet18模型
ret = rknn.load_pytorch(model='./resnet18.pt',
                        input_size_list=[[1,3,224,224]])
```

2.4 构建RKNN模型

API	build
描述	构建RKNN模型。
参数	**do_quantization**：是否对模型进行量化。默认值为True。
	dataset：用于量化校正的数据集。目前支持文本文件格式，用户可以把用于校正的图片（jpg或png格式）或npy文件路径放到一个.txt文件中。文本文件里每一行一条路径信息。如： a.jpg b.jpg 或 a.npy b.npy 如有多个输入，则每个输入对应的文件用空格隔开，如： a.jpg a2.jpg b.jpg b2.jpg 或 a.npy a2.npy b.npy b2.npy 注：量化图片建议选择与预测场景较吻合的图片。
	rknn_batch_size：模型的输入Batch参数调整。默认值为None，表示不进行调整。 如果大于1，则可以在一次推理中同时推理多帧输入图像或输入数据，如MobileNet模型的原始input维度为[1, 224, 224, 3]，output维度为[1, 1001]，当rknn_batch_size设为4时，input的维度变为[4, 224, 224, 3]，output维度变为[4, 1001]。 注： 1. rknn_batch_size只有在NPU多核的平台上可以提高性能（提升核心利用率），因此rknn_batch_size的值建议与核心数匹配。 2. rknn_batch_size修改后，模型的input/output的shape都会被修改，使用inference推理模型时需要设置相应的input的大小，后处理时，也需要对返回的outputs进行处理。
	auto_hybrid：是否开启自动混合量化调整精度或溢出。默认值为False，表示不进行调整。当模型经行量化时，开启自动混合量化会将低于指定阈值的余弦距离/欧式距离的op转为fp16计算(目前仅在w8a8量化下支持);当模型未进行量化时，开启自动混合量化会将超出fp16数值范围的op转为int16计算。 注： 1. 余弦距离和欧式距离的阈值可通过config接口中的 auto_hybrid_cos_thresh和auto_hybrid_euc_thresh设置。
返回值	0：构建成功。
	-1：构建失败。

举例如下：

```
# 构建RKNN模型，并且做量化
ret = rknn.build(do_quantization=True, dataset='./dataset.txt')
```

2.5 导出RKNN模型

通过本工具构建的RKNN模型通过该接口可以导出存储为RKNN模型文件，用于模型部署。

API	export_rknn
描述	将RKNN模型保存到指定文件中（.rknn后缀）。
参数	**export_path**：导出模型文件的路径。
返回值	0：导出成功。
	-1：导出失败。

举例如下：

```
# 将构建好的RKNN模型保存到当前路径的mobilenet_v1.rknn文件中
ret = rknn.export_rknn(export_path='./mobilenet_v1.rknn')
```

2.6 加载RKNN模型

API	load_rknn
描述	加载RKNN模型。 加载完RKNN模型后，不需要再进行模型配置、模型加载和构建RKNN模型的步骤。并且加载后的模型仅限于连接NPU硬件进行推理或获取性能数据等，不能用于模拟器或精度分析等。
参数	**path**：RKNN模型文件路径。
返回值	0：加载成功。
	-1：加载失败。

举例如下：

```
# 从当前路径加载mobilenet_v1.rknn模型
ret = rknn.load_rknn(path='./mobilenet_v1.rknn')
```

2.7 初始化运行时环境

在模型推理或性能评估之前，必须先初始化运行时环境，明确模型的运行平台（具体的目标硬件平台或软件模拟器）。

API	init_runtime
描述	初始化运行时环境。
参数	**target**：目标硬件平台，支持“rv1103”、“rv1103b”、“rv1106”、“rv1106b”、“rv1126b”、“rk3562”、“rk3566”、“rk3568”、“rk3576”和“rk3588”。默认值为None，即在PC使用工具时，模型在模拟器上运行。 注：target设为None时，需要先调用build或hybrid_quantization接口才可让模型在模拟器上运行。
	device_id：设备编号，如果PC连接多台设备时，需要指定该参数，设备编号可以通过“**list_devices**”接口查看。默认值为None。
	perf_debug：进行性能评估时是否开启debug模式。在debug模式下，可以获取到每一层的运行时间，否则只能获取模型运行的总时间。默认值为False。
	eval_mem：是否进入内存评估模式。进入内存评估模式后，可以调用eval_memory接口获取模型运行时的内存使用情况。默认值为False。
	async_mode：是否使用异步模式。默认值为False。 调用推理接口时，涉及设置输入图片、模型推理、获取推理结果三个阶段。如果开启了异步模式，设置当前帧的输入将与推理上一帧同时进行，所以除第一帧外，之后的每一帧都可以隐藏设置输入的时间，从而提升性能。在异步模式下，每次返回的推理结果都是上一帧的。 **（目前版本该参数暂不支持）**
	core_mask：设置运行时的NPU核心。支持的平台为RK3588 / RK3576，支持的配置如下： RKNN.NPU_CORE_AUTO：表示自动调度模型，自动运行在当前空闲的NPU核上。 RKNN.NPU_CORE_0：表示运行在NPU0核心上。 RKNN.NPU_CORE_1：表示运行在NPU1核心上。 RKNN.NPU_CORE_2：表示运行在NPU2核心上。 RKNN.NPU_CORE_0_1：表示同时运行在NPU0、NPU1核心上。 RKNN.NPU_CORE_0_1_2：表示同时运行在NPU0、NPU1、NPU2核心上。 RKNN.NPU_CORE_ALL：表示根据平台自动配置NPU核心数量。 默认值为RKNN.NPU_CORE_AUTO。 注：RK3576只有2个核心，因此不能设置NPU_CORE_2和NPU_CORE_0_1_2。
	fallback_prior_device：设置当OP超出NPU规格时fallback的优先级，当前支持“cpu”或“gpu”，“gpu”只有在存在GPU硬件的平台上有效。默认值是“cpu”。
返回值	0：初始化运行时环境成功。
	-1：初始化运行时环境失败。

举例如下：

```
# 初始化运行时环境
ret = rknn.init_runtime(target='rk3566')
```

2.8 模型推理

在进行模型推理前，必须先构建或加载一个RKNN模型。

API	inference
描述	对当前模型进行推理，并返回推理结果。 如果初始化运行环境时有设置target为Rockchip NPU设备，得到的是模型在硬件平台上的推理结果。如果没有设置target，得到的则是模型在模拟器上的推理结果。
参数	**inputs**：待推理的输入列表，格式为ndarray。
	data_format：输入数据的layout列表，“nchw”或“nhwc”，只对4维的输入有效。默认值为None，表示所有输入的layout都为NHWC。
	inputs_pass_through：输入的透传列表。默认值为None，表示所有输入都不透传。 非透传模式下，在将输入传给NPU驱动之前，工具会对输入进行减均值、除方差等操作；而透传模式下，不会做这些操作，而是直接将输入传给NPU。 该参数的值是一个列表，比如要透传input0，不透传input1，则该参数的值为[1, 0]。
返回值	results：推理结果，类型是ndarray list。

举例如下：

对于分类模型，如mobilenet_v1，代码如下（完整代码参考example/tflite/mobilent_v1）：

```
# 使用模型对图片进行推理，得到TOP5结果
outputs = rknn.inference(inputs=[img])
show_outputs(outputs)
```

输出的TOP5结果如下：

```
-----TOP 5-----
[ 156] score:0.928223 class:"Shih-Tzu"
[ 155] score:0.063171 class:"Pekinese, Pekingese, Peke"
[ 205] score:0.004299 class:"Lhasa, Lhasa apso"
[ 284] score:0.003096 class:"Persian cat"
[ 285] score:0.000171 class:"Siamese cat, Siamese"
```

2.9 评估模型性能

API	eval_perf
描述	评估模型性能。 模型必须运行在与PC连接的RV1103 / RV1103B / RV1106 / RV1106B / RV1126B / RK3562 / RK3566 / RK3568 / RK3576 / RK3588上。如果调用“**init_runtime**”的接口来初始化运行环境时设置perf_debug为False，则获得的是模型在硬件上运行的总时间；如果设置perf_debug为True，除了返回总时间外，还将返回每一层的耗时情况。
参数	**is_print**：是否打印性能信息，默认值为True。
	fix_freq：是否固定硬件设备的频率，默认值为True。
返回值	perf_result：性能信息（字符串）。

举例如下：

```
# 对模型性能进行评估
perf_detail = rknn.eval_perf()
```

2.10 获取内存使用情况

API	eval_memory
描述	获取模型在硬件平台运行时的内存使用情况。 模型必须运行在与PC连接的RV1103 / RV1103B / RV1106 / RV1106B / RV1126B / RK3562 / RK3566 / RK3568 / RK3576 / RK3588上。
参数	**is_print**：是否以规范格式打印内存使用情况，默认值为True。

返回值	**memory_detail**：内存使用情况，类型为字典。 内存使用情况按照下面的格式封装在字典中：
	{
	'weight_memory': 3698688,
	'internal_memory': 1756160,
	'other_memory': 484352,
	'total_memory': 5939200,
	}
	- 'weight_memory' 字段表示运行时模型权重的内存占用。
	- 'internal_memory' 字段表示运行时模型中间tensor内存占用。
	- 'other_memory' 字段表示运行时其他的内存占用。
	- 'total_model_allocation' 表示运行时的总内存占用，即权重、中间tensor和其他的内存占用之和。

举例如下：

```
# 对模型内存使用情况进行评估
memory_detail = rknn.eval_memory()
```

如examples/caffe/mobilenet_v2，它在RK3588上运行时内存占用情况如下：

```
=============================================================
            Memory Profile Info Dump
=============================================================
NPU model memory detail(bytes):
    Weight Memory: 3.53 MiB
    Internal Tensor Memory: 1.67 MiB
    Other Memory: 473.00 KiB
    Total Memory: 5.66 MiB

INFO: When evaluating memory usage, we need consider
the size of model, current model size is: 4.09 MiB
=============================================================
```

2.11 查询SDK版本

API	get_sdk_version
描述	获取SDK API和驱动的版本号。 注：**使用该接口前必须完成模型加载和初始化运行环境，且该接口只能在硬件平台RV1103 / RV1103B / RV1106 / RV1106B / RV1126B / RK3562 / RK3566 / RK3568 / RK3576 / RK3588上使用。**
参数	无。
返回值	sdk_version：API和驱动版本信息，类型为字符串。

举例如下：

```
# 获取SDK版本信息
sdk_version = rknn.get_sdk_version()
print(sdk_version)
```

返回的SDK信息类似如下：

```
=============================================
RKNN VERSION:
    API: 1.5.2 (8babfea build@2023-08-25T02:31:12)
    DRV: rknn_server: 1.5.2 (8babfea build@2023-08-25T10:30:12)
    DRV: rknnrt: 1.5.3b13 (42cbca6f5@2023-10-27T10:13:21)
=============================================
```

2.12 混合量化

2.12.1 hybrid_quantization_step1

使用混合量化功能时，第一阶段调用的主要接口是hybrid_quantization_step1，用于生成临时模型文件（<model_name>.model）、数据文件（<model_name>.data）和量化配置文件（<model_name>.quantization.cfg）。接口详情如下：

API	hybrid_quantization_step1
描述	根据加载的原始模型，生成对应的临时模型文件、配置文件和量化配置文件。
参数	**dataset**：见2.4 构建RKNN模型的dataset说明。
	rknn_batch_size：见2.4 构建RKNN模型的rknn_batch_size说明。
	proposal：产生混合量化的配置建议值。 默认值为False。
	proposal_dataset_size：proposal使用的dataset的张数。默认值为1。 因为proposal功能比较耗时，所以默认只使用1张，也就是dataset里的第一张。
	custom_hybrid：用于根据用户指定的多组输入和输出名，选取混合量化对应子图。格式为[[input0_name, output0_name], [input1_name, output1_name], ...]。默认值为None。 注：输入和输出名应根据生成的临时模型文件(<model_name>.model)来选择。
返回值	0：成功。
	-1：失败。

举例如下：

```
# 调用hybrid_quantization_step1 产生量化配置文件
ret = rknn.hybrid_quantization_step1(dataset='./dataset.txt')
```

2.12.2 hybrid_quantization_step2

用于使用混合量化功能时生成RKNN模型，接口详情如下：

API	hybrid_quantization_step2
描述	接收临时模型文件、配置文件、量化配置文件和校正数据集作为输入，生成混合量化后的RKNN模型。
参数	**model_input**：hybrid_quantization_step1生成的临时模型文件（<model_name>.model）路径。
	data_input：hybrid_quantization_step1生成的数据文件（<model_name>.data）路径。
	model_quantization_cfg：hybrid_quantization_step1生成并经过修改后的模型量化配置文件（<model_name>.quantization.cfg）路径。
返回值	0：成功。
	-1：失败。

举例如下：

```
# Call hybrid_quantization_step2 to generate hybrid quantized RKNN model
ret = rknn.hybrid_quantization_step2(
                model_input='./ssd_mobilenet_v2.model',
                data_input='./ssd_mobilenet_v2.data',
                model_quantization_cfg='./ssd_mobilenet_v2.quantization.cfg')
```

2.13 量化精度分析

该接口的功能是进行浮点、量化推理并产生每层的数据，并进行量化精度分析。

API	accuracy_analysis
描述	推理并产生快照，也就是dump出每一层的tensor数据。会dump出包括fp32和quant两种数据类型的快照，用于计算量化误差。 注： **1. 该接口只能在 build 或 hybrid_quantization_step2 之后调用。** **2. 如未指定target，并且原始模型应该为已量化的模型（QAT模型），则会调用失败。** **3. 该接口使用的量化方式与config中指定的一致。**
参数	**inputs**：图像（jpg / png / bmp / npy等）路径 list。
	output_dir：输出目录，所有快照都保存在该目录。默认值为'./snapshot'。 如果没有设置target，在output_dir下会输出： - simulator目录：保存整个量化模型在simulator上完整运行时每一层的结果（已转成float32）； - golden目录：保存整个浮点模型在simulator上完整跑下来时每一层的结果; - error_analysis.txt：记录simulator上量化模型逐层运行时每一层的结果与golden浮点模型逐层运行时每一层的结果的余弦距离（entire_error cosine），以及量化模型取上一层的浮点结果作为输入时，输出与浮点模型的余弦距离（single_error cosine），更详细的信息请查看error_analysis.txt文件。 如果有设置target，则在output_dir里还会多输出： - runtime目录：保存整个量化模型在NPU上完整运行时每一层的结果（已转成float32）。 - error_analysis.txt：在上述记录的内容的基础上，还会记录量化模型在simulator上逐层运行时每一层的结果与NPU上逐层运行时每一层的结果的余弦距离（entire_error cosine）等信息，更详细的信息请查看error_analysis.txt文件。
	target：目标硬件平台，支 持“rv1103”、“rv1103b”、“rv1106”、“rv1106b”、“rv1126b”、“rk3562”、“rk3566”、“rk3568”、“rk3576”和“rk3588”，默认为None。 如果设置了target，则会获取NPU运行时每一层的结果，并进行精度的分析。
	device_id：设备编号，如果PC连接多台设备时，需要指定该参数，设备编号可以通过“**list_devices**”接口查看。默认值为None。
返回值	0：成功。
	-1：失败。

举例如下：

```
# Accuracy analysis
ret = rknn.accuracy_analysis(inputs=['./dog_224x224.jpg'])
```

2.14 获取设备列表

API	list_devices
描述	列出已连接的RV1103 / RV1103B / RV1106 / RV1106B / RV1126B / RK3562 / RK3566 / RK3568 / RK3576 / RK3588。 注：目前设备连接模式有两种：ADB和NTB。多设备连接时请确保他们的模式都是一样的。
参数	无。
返回值	返回adb_devices列表和ntb_devices列表，如果设备为空，则返回空列表。

举例如下：

```
rknn.list_devices()
```

返回的设备列表信息如下：

```
************************
all device(s) with adb mode:
VD46C3KM6N
************************
```

注：**使用多设备时，需要保证它们的连接模式都是一致的，否则会引起冲突，导致设备连接失败。**

2.15 导出加密模型

该接口的功能是将普通的RKNN模型进行加密，得到加密后的模型。

API	export_encrypted_rknn_model
描述	根据用户指定的加密等级对普通的RKNN模型进行加密。 注：RV1103/RV1103B/RV1106/RV1106B/RK2118平台暂不支持。
参数	**input_model**：待加密的RKNN模型路径。
	output_model：模型加密后的保存路径。默认值为None，表示使用{original_model_name}.crypt.rknn作为加密后的模型名字。
	crypt_level：加密等级，有1，2和3三个等级。默认值为1。 等级越高，安全性越高，解密越耗时；反之，安全性越低，解密越快。数据类型为整型，
返回值	0：成功。
	-1：失败。

举例如下：

```
ret = rknn.export_encrypted_rknn_model('test.rknn')
```

2.16 注册自定义算子

该接口的功能是注册一个自定义算子。

API	reg_custom_op
描述	注册用户提供的自定义算子类。目前只支持ONNX模型。
参数	**custom_op**：用户自定义的算子类。用于用户需要自定义一个不存在于ONNX算子规范内的新算子。该算子的op_type推荐以“cst”字符开头，并且其算子类的shape_infer和compute函数需要用户自己实现。 注：custom_op算子类仅用于模型转换并生成带有自定义算子的RKNN模型，在设备端进行部署时还需要参考《RKNN SDK User Guide》的5.5章节。
返回值	0：成功。
	-1：失败。

举例如下：

```
import numpy as np
from rknn.api.custom_op import get_node_attr
class cstSoftmax:
    op_type = 'cstSoftmax'
    def shape_infer(self, node, in_shapes, in_dtypes):
        out_shapes = in_shapes.copy()
        out_dtypes = in_dtypes.copy()
        return out_shapes, out_dtypes
    def compute(self, node, inputs):
        x = inputs[0]
        axis = get_node_attr(node, 'axis')
        x_max = np.max(x, axis=axis, keepdims=True)
        tmp = np.exp(x - x_max)
        s = np.sum(tmp, axis=axis, keepdims=True)
        outputs = [tmp / s]
        return outputs

ret = rknn.reg_custom_op(cstSoftmax)
```

2.17 生成C++部署示例

API	codegen
描述	自动生成C++的部署示例。
参数	**output_path**：输出文件夹目录，用户可配置目录名称。
	inputs：填写模型输入的路径列表，允许不填。有效文件格式为jpg/png/npy，以npy文件为输入时，npy数据的维度信息应与模型输入的维度信息保持一致。
	overwrite：设为True时，会覆盖output_path指定目录下的文件。默认值为alse。
返回值	0：成功。
	-1：失败。

举例如下：

```
ret = rknn.codegen(output_path='./rknn_app_demo',
                   inputs=['./mobilenet_v2/dog_224x224.jpg'],
                   overwrite=True)
```